

Dependency Network Based Planning, Understanding the True Effects of Plans and Actions

Brian Drabble

Operations and Information Management Group (OIM),
University of Bradford, School of Management,
Emm Lane, Bradford,
West Yorkshire, BD9 1JL
bdrabble@bradford.ac.uk

Abstract

This paper describes an approach to intelligent planning which employs a dependency network model to reason with direct, n-order, cumulative and cascading effects. The dependency network is a cyclic graph of nodes (persons, organizations, locations, resources, concepts, etc.) connected by weighted arcs that define the type and strength of dependency between a node pair. The network can be employed as part of Mixed Initiative Planning (MIP) architecture or embedded as a constraint manager inside a fully automated planning architecture. The network can be employed to firstly evaluate both the intended, cumulative and cascading effects of actions and secondly to critique plans to identify their over dependence on a node or set of nodes. For example, repairing the electrical substation will restore power to the hospital and the traffic light sensors in the area. The choice of repair teams 1 and 2 to conduct the repairs means they are both dependent on two key roads to travel to the repair site. The paper provides details of both planning architectures, an overview of the dependency model and examples of its use on several scenarios.

Introduction

This paper describes an approach to intelligent planning which employs a dependency network model Drabble (2014, 2015) to reason with direct, n-order, cumulative and cascading effects. A dependency network is a cyclic graph of nodes connected by weighted arcs where the weighted values represent the strength of the dependency between a pair of nodes. For example, the dependency of a plane on a pilot, a telephone exchange on electrical power, etc. Using a language such as PDDL (2014) action descriptions can be developed which describe the direct effects of actions. However, PDDL and other language struggle to represent effects which occur over time or are triggered indirectly by one or more direct action effects. For example, the loss of

electrical power (EP) from a substation would impact any physical nodes directly *connected* to it and have a dependency on EP. While these direct links can be handled by the conditional effects extension to PDDL, the effect(s) of the loss of EP could propagate beyond the direct links to affect nodes 2, 3, 4 or more dependency links away from the impacted node. For example, if we have the dependency chain “*telephone exchange* → *transmission line 1* → *transfer station* → *transmission line 2* → *substation*” and a plan action directly affects the *substation* then what are the effects on the other nodes if any? This would need a series of embedded “*For Loop*” within the conditional effects to reason out to 2, 3, 4, etc. links but it would not be able to handle the different nodes and dependencies encountered. For example, if the *telephone exchange* has a backup EP generator then what is the impact on the exchange of the EP loss. If instead the node was a supermarket, then what is the impact. It would require a very large number of schemas to be developed to handle all the possible effects and maintaining them would be an even greater issue.

These effects at a distance are referred to as indirect or *n-order* effects of an action. N-Order effects can include indirect, cumulative and cascading ones and can be both desired and undesired. It would be extremely difficult for a user to write an action description in PDDL that was cable of reasoning over the complete set of situations in which an action could cause direct and n-order effects amongst a group of inter-dependent nodes such as the four above.

The remainder of the paper is as follows. The following three subsections provide details of the dependency network function with specific emphasis on how it supports reasoning with action preconditions and effects. The following sections provide further details of the dependency network and the use of the dependency network to support both an MIP and automated planning architecture. A summary and future research directions section is as provided.

Overview of a Dependency Networks Function

The purpose of using a dependency network is to handle reasoning with n-order effects and use its analysis in two different ways. Firstly, to provide a plan critiquing and constraint management function that aids a user within an MIP framework. Secondly, to provide a guidance and constraint management function within an automated planning architecture. An example of the former would be to identify that the plan has an over reliance on a single EP node hence making it vulnerable to execution changes. An example of the latter would be to identify that a partial plan was heavily dependent on using ground assets to move aid supplies¹ hence it would be better to select the “move by air” action to expand a high-level plan node.

The dependency network does not reason about nodes in a causal way, hence does not need a user to provide rules to identify changes in the network. Rather it identifies the type of relationship between a pair of nodes, its *strength* and the *consequence* (in terms of the change in the node’s output, capability², etc.) should the dependee node be degraded 100%. For example, EP substation (S1) is dependent on two transmission lines (T1 and T2) to supply it with EP. If T1 and T2 supply equal amounts of EP to S1 then both links $S1 \rightarrow T1$ and $S1 \rightarrow T2$ would be labelled *supply* and each would have a *strength* value of 5.0. If T1 was the primary source and T2 a backup, then the link values would be 10.0 and 0.0 respectively. The *consequence* value is used to propagate the positive or negative³ effect on a node’s output/capability, etc. as a result of the direct effect(s) of a plan. In the case of a 100% loss of the input provided by T1 then S1’s output would be specified by the input provided from T2. This example shows a simple pass through of a resource (EP) via S1 however, in most cases a node’s dependent inputs are either transformed or used to create another output or capability. For example, a machine to mix concrete is dependent on inputs of cement and water and on EP to power the machine. If any of these 3 dependent inputs were reduced by 100% then the output of concrete would be reduced to zero regardless of the level of input provided by the other two. This allows the dependency network to identify the *criticality* of a dependency and propagate the appropriate values through the network.

Dependency Networks for Effects Reasoning

The network allows for nodes to be linked in both directions with different relationships, strength and consequence values. For example, a customer of the cement plant could

be linked to the plant by a *supplies* relationship and the plant to the customer by a *financial support* relationship. Given the plant is the only one in the area the dependency of the customer on the plant is higher than plant on the customer as it has more than one customer. The same reasoning applies to the consequence values should either be reduced by 100%. The ability to handle loops between pairs of nodes or loops comprising multiple nodes allows the network to model feedback. This is where the initial direct plan effect on a node is magnified by a chain of consequence values that loops back to the initial plan effect nodes. For example, if a power station’s EP output is degraded 30% and this leads to a 40% drop in the amount of coal that can be delivered to the plant by rail this would lead to less EP being generated and hence even less coal being delivered. Eventually, the station’s EP output would reach zero due to the feedback loop between EP generation and its dependent supply of coal.

The ability to reason with direct and indirect effects, looping and feedback within a dependency models provides both human and automated planners with a range of different ways to affect a node(s) capability. Firstly, a plan effect could directly exploit the *vulnerability* of an actor node to “seize and arrest” or a physical node’s EP output to “repair”. Secondly, a plant’s output of concrete can be modified by exploiting the vulnerability of the dependee node(s) that supply cement, water or EP. As described earlier if one of these nodes is a critical dependency then only that one needs to be affected to halt concrete production. If one of these dependee nodes has dependee nodes itself, (e.g. the substation that provided EP to the transmission line that supplies the concrete plant) then it could be affected by a plan action. This allows planners to construct plans whose effects occur at one node in a dependency network but whose intended effect occurs at a node 1, 2, 3, etc. links away. The ability to identify and affect multiple nodes simultaneously provides planners with the capability to exploit cumulative and cascading effects. The feedback example shown in the previous section shows how a plan can cause an initial small effect on a node that can be magnified to achieve a much larger desired effect either directly or indirectly. This allows plans to be developed that are extremely well focused while looking to degrade network performance (insurgents, cyber-hackers) or provide the maximum return when improving performance (aftermath of Hurricane Katrina or post conflict reconstruction).

Dependency Networks for Precondition Reasoning

The network allows for the specification of plan preconditions that are based on a node or node output capability. If an action requires 500 gallons of fuel or access to a 2KV transmission line, etc. then these can be specified as an *initiating*, *maintaining* or *terminating* precondition. The

¹ There would be a corresponding n-order effect of a high dependency on the road segments, bridges, etc. that the vehicles travelled along.

² Capability, Experience, Skill and Knowledge are non-quantifiable outputs that have discrete levels “knowledge accountancy expert”

³ Actions such as Degrade, Dismantle, Dislocate reduce a node’s outputs where Repair, Reinstall, Re-initialize increase a node’s outputs.

dependency network provides search capabilities to identify whether this can be supplied by a single network node or from contributions provided by multiple network nodes. It also provides feedback on any modifications that may be needed in the plan to provide the required level of input. For example, the start time window of Action 1 must be restricted from the range 3 -6 to 3 in order for it to return sufficient resources needed by the requesting action. Restrictions can also be placed on open plan variables to stop them using a specific dependency node or nodes. Preconditions can identify a specific output or a capability level that a node must possess. For example, a truck has the capability to provide ground transport or the capacity of the truck to carry 2500lbs of cargo can be shared by multiple transport actions. The network model tracks changes to node outputs/capabilities as action that affect the node(s) are inserted or ordered within a plan. It does this by asserting a *concept* node into the network to represent a high level or primitive action and links it to the node(s) that satisfy its direct precondition or are impacted by an effect.

Dependency Networks

A dependency network comprises multiple intra and inter-dependent networks that reflect the Political, Military Economic, Infrastructural, Informational and Social (PEMSII) (2016) aspects of the domain being modelled. Examining the interdependencies between nodes helps understand how the combined networks function, the most important nodes overall and most importantly what effects a plan will truly have in a domain. The importance of a node can be identified through a range of different measures:

- The cumulative weighted dependency that other nodes have on it (**dependent** analysis)
- The cumulative weighted dependency that a node has on other nodes (**dependee** analysis)

Dependency can be stated directly between any pair of nodes in the model and reflects the strength of the dependency that one node has on another. The strength of a dependency is measured using a scale of 1 – 10 where 1 reflects a weak dependency and 10 a critical one. The direct dependencies are then used to define a transitive dependency between two nodes linked by one or more intermediary nodes. For example, the direct dependency of the concrete plant on the transmission line that results in the transitive dependency of the plant on the EP substation. It is often the cumulative transitive dependencies that define the importance of a node and not the usually smaller set of direct pairwise dependencies. Athena⁴ provides various network analysis algorithms to generate a dependee and

dependent score for each node in a network. The dependent or dependee score for a node allows it to be ranked against other user selected nodes to identify its importance. The scores can also be used to compare nodes based on the ratio of their dependent or dependee scores. A screenshot from Athena is shown in Figure 1.

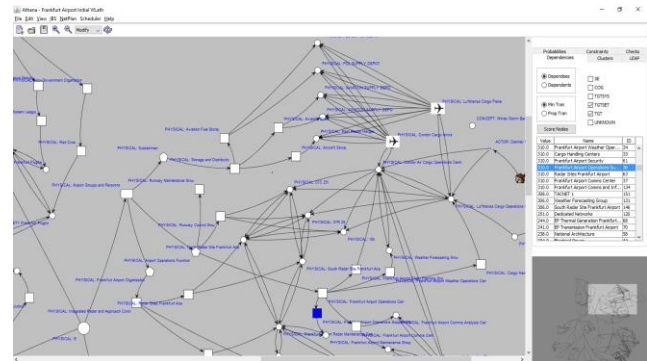


Figure 1: Dependency Model of Frankfurt Airport

Figure 1 shows a model of the cargo operations at Frankfurt Airport in terms of the dependencies between the air cargo companies (Lufthansa and Condor) and the groups and infrastructure of the airport. The analysis displayed middle right of Figure 1 shows that the *Frankfurt Operations Support Center* (the highlighted blue node) with a score of 310 has almost 25% more importance to the Airport functions than the airports *Electrical Power Transmission Networks* with a score of 241. While the *Support Center* has the greatest dependency score, analysis of the consequence of losing either of these shows that a 100% loss of the *Transmission Network* results in a far greater number of nodes being affected and by far greater percentage change. Figure 2 shows the consequence analysis for the change to the *Transmission Network* as the percentage change to the nodes impacted (displayed middle right).

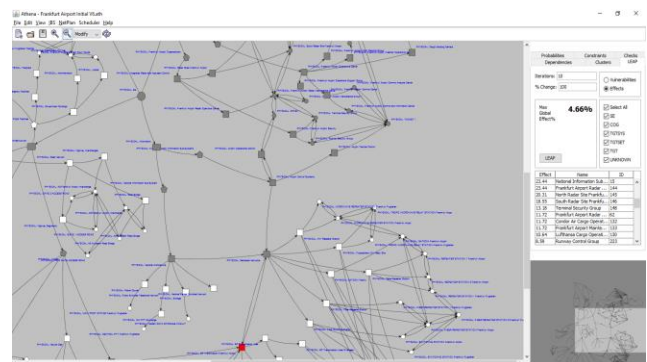


Figure 2: Example Consequence Analysis

A human or automated planner can use the dependency and consequence scores to identify which parts of the model their plan must try to avoid directly affecting and if not what level of change could be acceptable. Secondly, it identifies which dependee nodes of the *Transmission Net-*

⁴ Athena is the name of the dependency analysis toolkit, system.

works should not be affected directly or indirectly. Thirdly, it aids the planner in prioritizing tasks should weather or other issues affect airport operations. If resources are limited then repairing, re-energizing the *Transmission Network* will have greater impact on operations than repairing damage to the functions of the *Operations Center*.

A planner can also use the model to identify the capabilities and dependencies of nodes specified in plan pre-conditions or effects. Figure 3 shows the specification of information associated with the nodes representing the *Condor Air Fleet* and their *Main Hanger*. The three boxes shown to the left define the capabilities of the Hanger to provide aircraft storage and repair facilities and its dependency on high voltage EP. The top window on the right describes the link from the Cargo Fleet to the Hanger describing the dependencies of the Hanger and the capabilities of the Cargo Fleet. The base of the window shows the definition of the mapping of the dependency of the Hanger on the Company. In this case, it is a financial one in that the Cargo Fleet has a contract with the Hanger. This also shows the dependency of the Main Hanger on High Voltage EP which needs to be supplied via another node. Hence even though there is a direct dependency between the hanger and the aircraft it cannot be used by the planner to affect the supply of EP to the hangar. The lower window describes the link from the Cargo Fleet to the Main Hangar. This shows the obvious mapping of the hanger's capabilities for repair and storage and the Cargo Fleets dependency on these capabilities to operate.

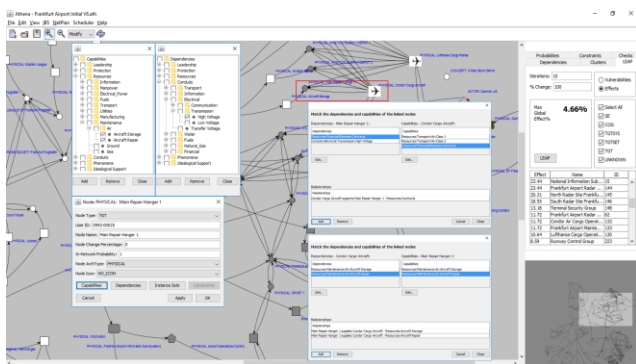


Figure 3: Example Attribute and Link Definitions

The dependency network also provides other additional insights that can be exploited by the planner. This aids in deciding the priority of goals/tasks and the plan(s) that should be developed to address them. Below is a list of the analysis capabilities provided.

Effects Analysis: Ranks nodes based on the total change in the network if they were degraded 100%. Identifies which nodes have the greatest effect across the network.

Vulnerability Analysis: Ranks nodes based on how vulnerable they are in terms of their strength of dependencies. Identifies which nodes are “easiest” to affect indirectly.

Node Vulnerability: For a selected node, it identifies the percentage capability change if the dependee node is degraded by 100% and is applied to each node dependee.

Critical Node Vulnerabilities: For a selected node identifies if there is a dependee node that can degrade the selected nodes capability by 100% and the percentage change to the dependee node to achieve the 100% degradation.

Cluster Analysis: Identifies independent sub-networks whose links all have strength equal to or greater than a user specified value.

The following sections provide details of the use of the dependency network to support both an automated and MIP based view of plan generation.

Mixed Initiative Planning Architecture

When employed as part of a MIP based planning architecture the dependency network supports three primary functions: Task Specification, Plan Development and Plan Critiquing. Each is described in the following sections.

Task Specification

Task Specification aids the user in deciding what nodes should be the focus of the plan and the desired network behavior to be achieved. As stated earlier a node's output/capability can be affected directly or alternatively indirectly by means of changes to one or more of its dependee nodes. Task specification also aids in identifying any nodes that should not be affected by the plan or the effect on them should be minimized. For example, a plan to replace one of the main EP transmission lines should not impact the airport operations center by more than 10%. Once the user has identified the node(s) to be affected a portal inside of Athena allows the user to specify these nodes, the level of effect on the desired node, direct or indirect influence and any resource constraints. Figure 4 shows an example set of specifications for a raid on a suspected insurgent safe house by a Special Forces Team. The primary goal is to Raid the safe house and this has already been specified. The portal shows the specification of a secondary goal to create checkpoints around the area to control ingress and egress of people, vehicles, etc. The task specification is then passed to NETPlan planner.

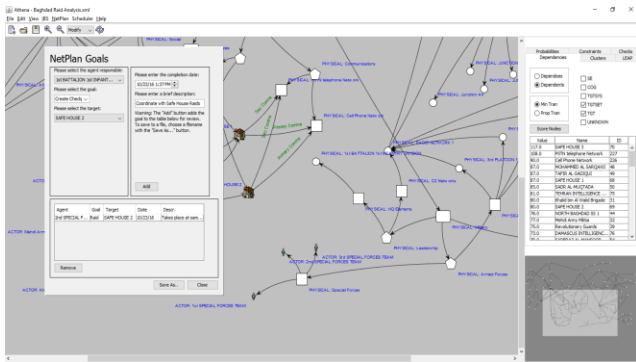


Figure 4: Example Task and Goal Specification

Plan Development

NETPlan provides the user with the ability to develop a plan based on the tasks, constraints, etc. specified. Figure 5 shows the plan developed for a raid on two safe houses and the construction of the checkpoints to support it. Along the top of Figure 5 are a series of green diamonds representing sub-goals and deadlines by which certain aspects of the plan should be completed. NETPlan uses a timeline based approach to plan representation where each row describes the actions assigned to the resource. Resource rows can be related to one another hierarchy and can be “rolled up” and hidden if desired. A user uses NETPlan to manually construct plans using the timeline based representation. NETPlan ensures that all time and resource constraints are maintained and warns the user if any are threatened. The user can then decide to alter the plan themselves or delegate this to NETPlan to re-order or move actions on the resource timeline accordingly.

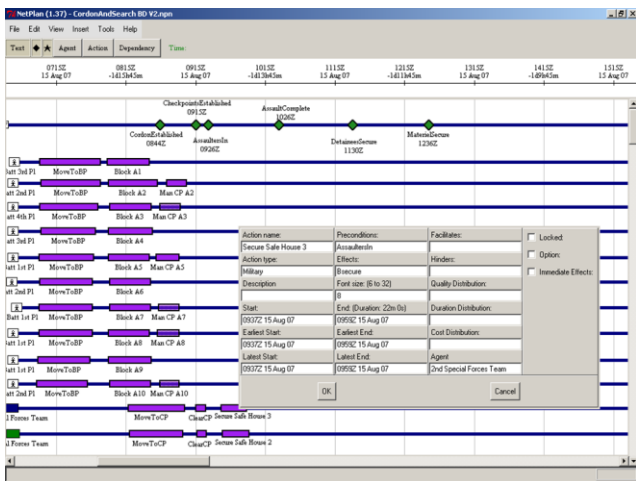


Figure 6: Example Plan Development with Network Support

Any action, resource, goals or sub-goal can be selected and information regarding the item displayed via a portal. Figure 6 shows the timing, resource, precondition, effect, etc. information for the action to *Secure Safe House 3*. The user

can use Athena to test different partially or fully developed plan options to understand what effects (desired and undesired) the plan has on the network. Essentially, the network acts as the initial state of the planning problem and the user is looking for guidance and feedback as to how well the plan matches against the goals specified. Based on Athena’s analysis the user could decide to re-assign tasks to a different resource due to a high dependency score, add an additional resource to reduce dependency or limit the impact of an effect, etc.

Plan Critiquing

Once the user has developed the plan to the required level of specification it can be *imported* into the dependency network. This is achieved by mapping the actions to a series of arcs that link the resource conducting the actions and the focus of the action. For example, the action “*Raid and Secure Safe House 3 Second Special Forces Team*” would be translated into an arc between the *Second Special Forces Team* and the *Safe House*. In order to simplify the mapping process Athena provides a series of ontology based mapping tools to identify matching items in the plan and network. Figure 7 shows the Athena portal that displays the mapping of the nodes in the plan to those in the network. A previous step had already dealt with the mapping of resource names. The second line of the table shows the entry in the Athena Target column entitled *Junction A1* whereas the corresponding NETPlan target is just *A1*. Based on information such as the capability of a road junction can be degraded by blocking or reducing its flow, Athena was able to deduce this was the probable mapping. If an incorrect mapping is generated, then the user can override it and Athena updates the ontology appropriately.

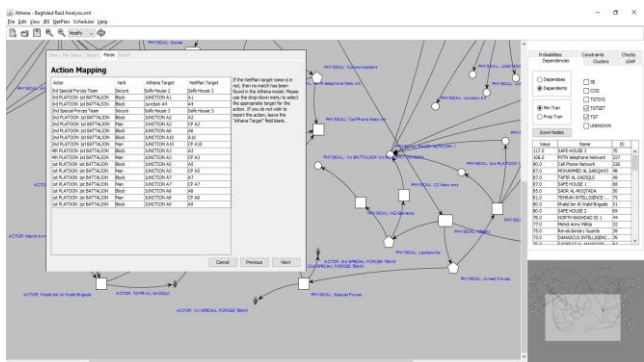


Figure 7: Mapping from Plan to Network Terms

Once all mappings are correct the plan can be imported into the network by means of additional nodes (if the resource or focus of effect was not already in the network) and links. Figure 8 shows the insertion of the links between the *Special Forces Team* and the *Safe house* and the subsequent analysis of the plan effects. The analysis pane shown middle right shows the 100% degradation of the safe house and the indirect effects on the other nodes listed.

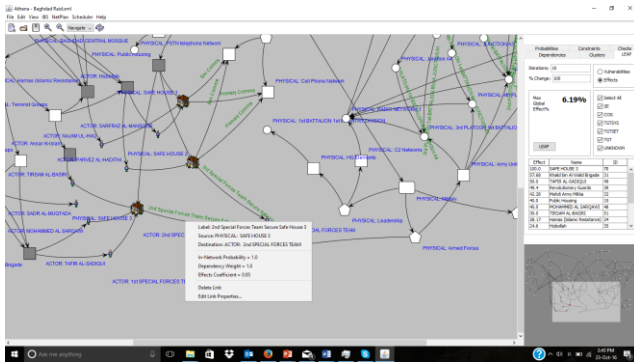


Figure 8: Action Insertion and Analysis

This is good plan as it targets the safe house and does not cause wide spread secondary effects. However, re-running the dependency analysis identifies that the node with the greatest dependency is now *North Baghdad SS1* hence the question is why. Figure 9 shows the dependent analysis for SS1 and it identifies that one of its main dependents is the radio network that the forces manning the roadblocks and checkpoints are using. As the forces are strongly dependent on the radio network and the plan is strongly dependent on the forces then the plan is strongly dependent transitively on the radio network. The importance of SS1 increased due to the dependencies on it from the resources employed in the plan which is something that the plan developer knew nothing about. Identifying potential points of dependency introduced via a plan helps the user develop plans which are most robust against events (weather, etc.) or the actions of others. In this case selecting unit's dependent on different radio networks or tasking a unit to protect SS1 during the raid to ensure continuity of EP for the radios.

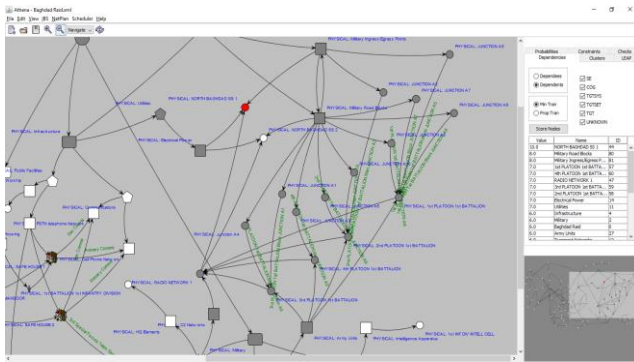


Figure 9: Plan Critiquing to Identify Weaknesses

The critiquing process can also identify unintended consequences that the planners may not be aware of due to their training, location or tasking. Dependencies change at different time of the day, in different locations, etc. One of the effects of the raid plan was is to reduce traffic flow in the locality of the safe houses. The same roads and junctions are also dependees of the local mosque. Hence if the raid is launched at 2.00am then there are few issues however, if it

is launched at noon on Friday when people are attending Friday prayers then the unintended consequences of the plan would be potentially significant. The ability to alert a human Special Forces planner as to the sensitivity of the launch time of the plan, was seen as a significant advantage. Their normal training does not involve looking for such interactions and potential conflicts.

Automated Planning Architecture

When employed as part of an automated planning architecture the dependency network can serve the same three primary functions of the MIP approach and provide the ability to guide the selection of options for schema selection, variable assignment, precondition satisfaction, etc. Each is described in the following sections.

Schema and Variable Selection

The ability to select the appropriate schemas and variables that best match the needs of a plan is one of the key aspects of any automated planning approach. The dependency model was incorporated into an automated planner called Minerva to provide direct feedback to the planning algorithm. This raises several important research questions. The first was how to describe the task to the planner in a form that captured all the relevant information. Below is a task description that was developed for Minerva.

task "SOF Raid";

vars ?verb = ?(or ?(type dverbs) ?(type rverbs)),
?noun, ?output, ?constraint, ?action, ?mechanism;
nodes

1 start;

2 finish,

3 action {{Degrade SF3 IC 100 "seizure and occupation"}}{{V=Degrade, N=AD, O=TWK and C=50}} {V=Degrade, N=IG1, O=Weapons and C=50}};

orderings 1 → 3, 3 → 2;

end_task;

In the case the task is to Degrade SF3's (Safe house 3) information conduit (IC) output by 100% and the method to be used is *seizure and occupation*. Tasks can be specified in terms of a method or in terms of the resource assigned. For example, instead of specifying the *seize* method the task could specify *SOF via Helicopters* and have Minerva select the method. The remainder of action 3 is a set of Meta-data that indicates the path through the dependency networks the plan effects should follow. In this case the aim of seizing the safe house is to degrade by 50% the locations from which node AD (a person or group) can transfer their working knowledge (TWK) and so degrade the ability of insurgent group IGI to produce weapons by 50%. Essentially by removing SF3 as an information conduit the person AD cannot train people and this should impact on

weapons production. The orderings information identifies the task occurs between the start and end of the plan. Minerva uses the same schema layout and structure as O-Plan (1999) but it has been heavily modified to deal with the interactions with the dependency network.

The action *{Degrade SF3 IC 100 “seizure and occupation”}* needs to be decomposed into a lower level action. Similar to O-Plan, Minerva uses a Partial Order Causal Link approach to plan generation. The following schema show one possible way to decompose the task description and introduces some of the additional structures that were added to allow interaction with the dependency network.

```

schema ConductRaidtoDegradeLocation;
  vars $verb, $percentage, $transport1, $transport2,
    $mechanism, $checkpoint,
    $checkpointsrequired = ?{type integer}
    $RaidLocation = ?{type physical}
    $SupportingUnits = ?{type actor, ?{or ?{capability
      “Conduct Ground Operations”}, ?{experience
      “Security Operations”}}},
    $RaidingForces = ?{type actor ?{or ?{capability
      “Raid Location”}, ?{experience “Raid Loca-
      tion”}}};
  expands {$verb $RaidLocation $output $mechanism
    $percentage};
  nodes
    1 action $checkpointsrequired Iterate action {Move
      $SupportingUnits $transport1 $checkpoint},
      {Establish_and_Man $checkpoint} for
      $checkpoint over ?{type poten-
      tial_checkpoints},
    2 action {Move $RaidingForces via $transport2 to
      $RaidLocation},
    3 action {Raid $RaidLocation by $RaidingForces
      $percentage};
  orderings 1 → 3, 2 → 3;
  conditions
    compute_condition ?{fn_ask “number of
      check points required” ?{or undef ?{type
      integer}} } = $checkpointsrequired,
    compute_condition ?{fn_mult $check-
      pointsrequired 5} = $RequiredManpower,
    compute_condition ?{output “manpower”
      $RequiredManpower units} = $Sup-
      portingUnits at begin_of self,
    compute_condition ?{output “manpower”
      20 units} = $RaidingForces at begin_of
      self,
    unsupervised {Location $RaidingForces} =
      $RaidLocation at 3,
    supervised {Status Security_Cordon $raid-
      location} = complete at 3 from 1;

```

```

effects      {Status Security_Cordon $RaidLocation} =
             complete at 1;
end_schema;

```

The schema *vars* structure describes the schema variables that are used in the schema definition. There are various restrictions or constraints placed on the schema variables. For example, *\$RaidLocation* has to be a physical type node in the network and *\$RaidingForces* need to be an actor (person, groups, etc.). In addition, *\$RaidingForces* need to possess the *capability* or *experience* to conduct a Raid at a location. A node can be assessed in terms of its output if it is quantifiable (EP, flow rate, etc.) or in terms of possessing a capability, experience, skill or knowledge to a specified level if it is not quantifiable (supervision, expert knowledge of painting, etc.). This structure was developed to augment the simple *capability* and *dependency* framework in Figure 3 to correctly capture these non-quantifiable aspects of nodes in a form that can be used by Minerva. The *expands* pattern matches the task specification in the task schema if the types and constraints on the schema variables match. The Meta-data from the task schema is simply tagged to this schema for checking the plan does what it is tasked to do.

The *nodes* statement describes the next level of the plan hierarchy. In this case the raid action comprises three sub-actions. The first is to insert a “*Move \$SupportingUnits \$transport1 \$checkpoint*” and “*Establish and Man \$checkpoint*” action for as many checkpoints that are required in the plan. This can be specified by the user or identified by Athena based on the level of impact the plan is desired to achieve. The remaining two actions move the raiding forces to the location via a transport and then raid the location. If at the end of the schema expansion a plan schema variable has not been assigned a value, then it is converted to a Plan State Variable (PSV). Minerva tags the PSV with all of the appropriate type and constraint information so it can be assigned later. For example, if there is a high dependency score on the helicopter nodes in the network then Minerva may choose to assign *\$transport2* to ground based vehicles so as to not increase the helicopter dependencies.

Precondition and Effect Analysis

The *conditions* structure contains multiple *compute_condition* statements that result in queries to the dependency network. These are preconditions the dependency network needs to address. The first identifies the number of checkpoints required and the second identifies the total amount of manpower required assuming 5 persons per checkpoint. The number of raiders is fixed at 20. Hence the dependency network needs to find one or more nodes that have the required attributes as defines in the *vars* statement and the required levels of output as described in the preconditions. The dependency network may return “No” as it

cannot find the required node(s), “**Yes**” here is a list of candidates and based on their dependency score they should be selected as follows or it could return “**Maybe**”. In the case of **Maybe** it means there are changes to the outputs of certain nodes caused by already asserted plan effects and if the following changes are made to the plan then the resources can be provided. For example, if an action is ordered earlier in the plan it will return sufficient resources to allow the schema to be selected. It is essentially a dependency based Modal Truth Criterion based on the state of the dependency network. Minerva can assess each of the **Maybe** options and select the one that best suits the current context. Once selected it creates a link between the action and the selected dependency node(s) to ensure the correct output level is provided. If the link is broken it is treated in the same way as a broken plan precondition.

The effects structure describes the changes to a node output/capability directly caused by the action. The schema below would be a candidate to expand the *Establish and Man Checkpoint* action described in the previous schema.

```

schema EstablishAndManCheckpoints;
  vars $Checkpoint = {?type physical},
        $AssignedUnit = {?type actor};
  expands
    Establish_and_Man $Checkpoint $AssignedUnit;
  only_use_for_effects
    {Checkpoint $Checkpoint} = established;
  condition
    compute_condition ?{output “manpower” 5 units}
      = $AssignedUnit during self,
    compute_condition ?{output “electrical power”
      500 units = $Checkpoint during self;
  effects
    ?{output ?{change “traffic flow” -50 percent-
      age}} = $Checkpoint during self;
  time_windows = 2 hours;
end_schema;

```

The effect statement states the action will result in a 50% reduction in traffic flow rate during the duration of action (*during self*). This means after the action is complete the effect stops and the flow rate should increase by 50% unless another action’s effects impact it directly or indirectly at the same time. The same type of reasoning also applies to the *condition* statements. This schema shows a change in the level of the assigned manpower of 5 units and this will only be allocated from the dependency node identified for the duration of the action. Afterwards the manpower is reallocated back to the unit. The same applies to the need for 500V units for the checkpoint which is taken from the EP network as load. Once the action is over the load can be returned. The use of the *self* statement in a *compute_condition* informs Minerva that the resource is allocated and then deallocated back to the dependency node.

Athena tracks changes in the network caused by plan effects and alerts the Minerva if changes threaten a precondition, schema choice, etc. so the planner can take appropriate actions. In this example the plan developed was to man 8 road blocks selected by Athena based on the local street layout and to raid 2 safe houses. Minerva inserted the second safe house raid as raiding just one did not achieve the desired impact on the *TWK* output of *AD*. It also added a third raid to avoid that being used as an alternative when the first two were lost. It selected a helicopter insertion method for the raiders to minimize disruption and selected a 0500am start to avoid other collateral issues.

Summary and Future Work

This paper described the use of a dependency based network model to support both an MIP and automated planning architecture. The model provides both human and automated planners with the ability to truly assess the impact of action effects in terms of the both direct and n-order ones. This allows plans to be constructed that allow goals to be achieved by directly affecting a node’s output or by affecting one or more of its dependees. The approach has been applied to a variety of scenarios including countering insurgencies, disaster and emergency management, logistics, etc. The current Cassandra and Minerva systems are being extended to allow reasoning with unknown nodes (those not instantiated to specific value) and how they could be used to guide the planner as to which aspects of the planning problem to focus on next. Other extensions include reasoning with probabilities with respect to the presence of a link or node in the network and the use of missing dependency information to guide schema and variable selection.

References

- PDDL. 2016, Planning Domain Definition Language. https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language
- PEMSII. 2016, Political, Economic, Military, Social, Infrastructural and Information model of a system, entity or state http://pmesii.dm2research.com/index.php/Main_Page
- O-Plan: Tate, A., Dalton, J. and Levine, J. 1999. Multi-Perspective Planning - Using Domain Constraints to Support the Coordinated Development of Plans, O-Plan Final Technical Report AFRL-IF-RS-TR-1999-60. .
- Drabble, B. 2015. Dependency Based Analysis to Support Robust Schedule Generation, Final Report, Naval Postgraduate School, Contract N00244-14-1-0056 NAVSUP, Monterey, CA, USA.
- Drabble, B. 2014. Modeling C2 Networks as Dependencies, understanding what the Real Issues Are. In Grant, T. J., R. H. P. Janssen, and H. Monsuur., Network Topology in Command and Control: Organization, Operation, and Evolution. pp125-151, accessed May 21, 2015. DOI: 10.4018/978-1-4666-6058-8. IGI Global Press.